Dkt. 0655/64696

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application of : Richard H. HARVEY

Serial No. : 09/827,738 Group Art Unit: 2171

Date Filed : April 6, 2001 Examiner:

For : DIRECTORY SEARCHING METHODS AND SYSTEMS

1185 Avenue of the Americas
New York, N.Y. 10036

Assistant Commissioner for Patents
Washington, D.C. 20231

CLAIM FOR PRIORITY UNDER 35 U.S.C. §119

Applicant hereby claims priority under 35 U.S.C. §119. Applicant hereby transmits a certified copy of the following priority application:

Application No.	Filed in Australia
PQ 6785	April 7, 2000

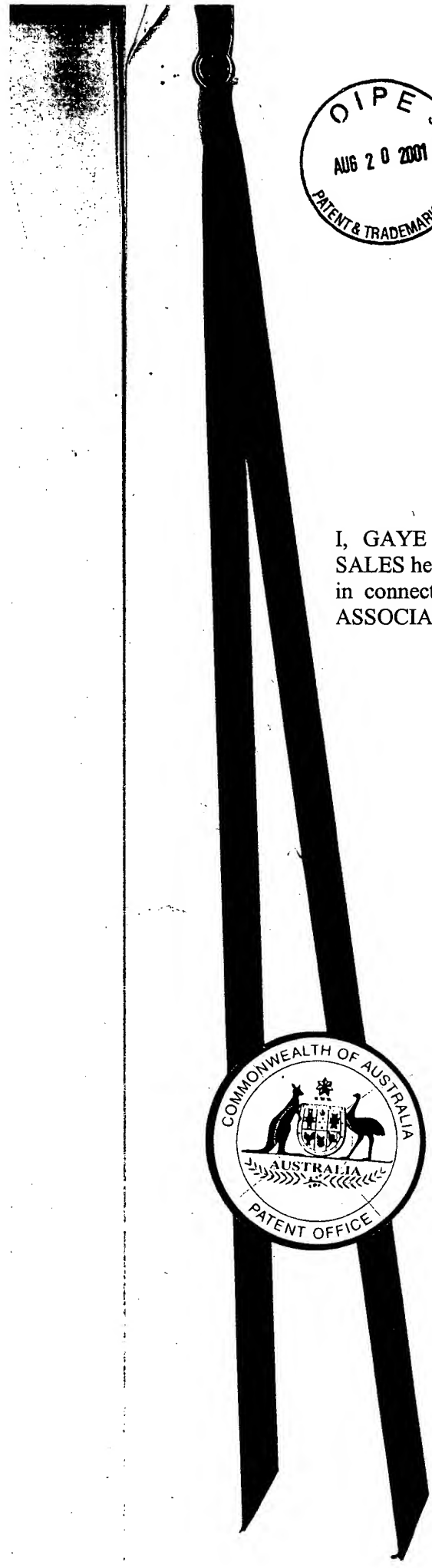
Respectfully submitted,

RICHARD F. JAWORSKI
Registration No. 33,515
Attorney for Applicant
Cooper & Dunham LLP
Tel.: (212) 278-0400

I hereby certify that this paper is being deposited this date with the U.S. Postal Service as first class mail addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231.

Aug. 16, 2001
Richard F. Jaworski Date
Reg. No. 33,515

0655/64696
s/n 09/827,738



Patent Office
Canberra

I, GAYE TURNER, TEAM LEADER EXAMINATION SUPPORT AND SALES hereby certify that annexed is a true copy of the Provisional specification in connection with Application No. PQ 6785 for a patent by COMPUTER ASSOCIATES THINK INC. filed on 07 April 2000.

WITNESS my hand this
Twentieth day of July 2001

GAYE TURNER
TEAM LEADER EXAMINATION
SUPPORT AND SALES

CERTIFIED COPY OF PRIORITY DOCUMENT

P/00/009 28/5/91
Regulation 3.2

AUSTRALIA

Patents Act 1990

PROVISIONAL SPECIFICATION

Invention title **DIRECTORY SEARCHING APPARATUS AND METHOD(S)**

The invention is described in the following statement:

DIRECTORY SEARCHING APPARATUS AND METHOD(S)

FIELD

The present invention is directed to a method(s) and apparatus for searching directory service databases and systems. In particular, but not exclusively, the present invention has application to systems and directories, which implement or perform X.500 or LDAP services in a relational database. The present invention is also directed to table structure and methods of operation of a database application.

PRIOR ART

X.500 is the International Standard for Electronic Directories [CCITT89 or ITU93]. These standards define the services, protocols and information model of a very flexible and general purpose directory. X.500 is applicable to information systems where the data is fairly static (e.g. telephone directory) but may need to be distributed (e.g. across organisations or countries), extensible (e.g. store names, addresses, job titles, devices etc.), object oriented (i.e. to enforce rules on the data) and/or accessed remotely.

Relational Database Management System

A Relational Database Management System (RDBMS) provides facilities for applications to store and manipulate data. Amongst the many features that they offer are data integrity, consistency, concurrency, indexing mechanisms, query optimisation, recovery, roll-back, security. They also provide many tools for performance tuning, import/export, backup, auditing and application development.

RDBMS are the preferred choice of most large scale managers of data. They are readily available and known to be reliable and contain many useful management tools. There is a large base of RDBMS installations and therefore a large amount of existing expertise and investment in people and procedures to run these systems, and so data managers are looking to use this when acquiring new systems. Most relational database products support the industry standard SQL (Structured Query Language).

There has also been a move towards Object Oriented systems, which provide data extensibility and the ability to handle arbitrarily complex data

items. In addition, many corporations and government departments have large numbers of database applications, which are not interconnected. Data managers are looking for solutions which enable them to integrate their data, and to simplify the management of that data. X.500 and its associated standards provide a framework and a degree of functionality that enables this to be achieved. The fact that X.500 is an international standard means that data connectivity can be achieved across corporations and between different countries.

One problem, therefore, is to address the need of data managers and implement X.500 with all the flexibility of object-oriented systems but using an SQL product so that it can achieve the scalability and performance inherent in relational systems coupled with the stability, robustness, portability and cost-effectiveness of current SQL products.

There have been a number of attempts at solving the above problem and over a considerable period of time. Some solutions to the problem noted above are disclosed in PCT/AU95/00560, now granted as Australian Patent No.712451

Despite the various solutions disclosed in PCT/AU95/00560, customers continuously seek other and even speedier solutions.

It has been discovered that there is a further improvement possible involving complex data types and, in particular, current searching techniques used to search for them.

The present invention, in particular, is directed to providing further improvement(s) by addressing a number of outstanding problems; namely

1. that directories generally treat attribute values in a holistic way,
2. that the directory query language, such as X.500 or LDAP, has not, until recent extensions, been able to express a search query in a manner that could find an attribute based on one of its internal components,
3. that the underlying database language, such as SQL, can only match on entire values or substrings within values, and has no mechanism for

matching against values that are made up of components, each possibly having a different type, and

4. that the previous systems, such as that disclosed in PCT/AU95/00560 are based on a metadata design, which in turn is based on a single index of single attribute values.

SUMMARY OF INVENTION

The present invention has as its object to address the outstanding problems noted above and to provide an improved directory services (such as X.500 and / or LDAP) system, method of interrogation and / or operation thereof.

To this end, the present invention provides, in one aspect, a method of arranging data in a database, the method including the steps of: providing a first table having one row for each entry, and being adapted for storing the data, and providing a second table having one row for each component of the data, and being adapted for storing data components.

Preferably, the data is a structured data type or a string data type.

In another aspect, the present invention provides a method of searching a database for a given data entry, the database having data stored therein, and having a first table having one row for each entry, and being adapted for storing the data, and a second table having one row for each component of the data, and being adapted for storing data components, the method including the steps of:

determining a component of the given data entry,

executing exact or initial matching on the second table in order to locate the component and

returning the given data entry matching the component located.

In accordance with another aspect, the present invention provides a database having a data storage arrangement, including

a SEARCH table containing at least one row having columns of the form, (EID, AID, VID, Norm), where, EID identifies the object to which the value belongs, AID identifies the attribute type of the value, and VID identifies one of a possible number of attribute values in the one entry, and

a SUBSEARCH table also containing zero or more rows having columns of the form (EID, AID, VID, CID, Norm), characterised in that

the SUBSEARCH table includes an additional CID column representing a component identifier.

In a still further aspect, the present invention provides a database having a data storage arrangement, including

a first table directed to hierarchy which defines the relationship between objects, the first table being arranged according to one row per object,

a second table directed to objects which defines the value(s) within each object, the second table being arranged according to one row per value, and

a third table directed to selected component(s) of values, the third table being arranged according to one row for each component of each value.

Preferably, the database is a part of a directory services system, such as x.500 or LDAP services system, such as a x.500 or LDAP directory services system.

In one particular application of the present invention, there is provided a method of storing a X.509 certificate in a database, where a check sum or fingerprint is stored. Furthermore, searching for a binary data type stored in a database, may include searching for the checksum or fingerprint.

In essence, the present invention involves the storing and/ or searching for the data types using one of the components contained in the data types. In effect, the present invention is based on the realisation that the extensible nature of the metadata design can be used to address the problem of searching complex data types by adding an extra column representing what we call a component identifier. Alternatively, a new table can be added, the new table including various information used for searching as well as the additional component identifier column. This develops what is referred to as a SUBSEARCH table which stores components of values and addresses the problems above by allowing the individual components to be searched.

In a metadata design, an example SEARCH table (as disclosed in PCT/AU95/00560) contains rows of the form, (EID, AID, VID, Norm), where,

EID identifies the object to which the value belongs, AID identifies the attribute type of the value, VID identifies one of a possible number of attribute values in the one entry, and Norm contains the syntax normalised value.

In one embodiment, what is stored in the SUBSEARCH table is substantially similar in structure to the SEARCH table, (although the information may be selected according to the particular application/use) but including a column (say CID) representing a component identifier.

At a more conceptual level, in the disclosure of PCT/AU95/00560, the tables looking after hierarchy, are NAME, DIT and TREE. These tables are arranged according to one row per object. The tables looking after objects, are SEARCH and ENTRY. These object tables are arranged according to one row per value. In the present invention, however, the SUBSEARCH table has one row for each component of each value. Furthermore, in the disclosure of PCT/AU95/00560, every object has a corresponding row in the hierarchy tables and every entry had a corresponding row in the object tables, but in the present invention, the SUBSEARCH table preferably stores only those values as thought desirable, such for performance, components of complex data. Other components stored in the SUBSEARCH table are those selected for reasons of manageability. In other words, in the SUBSEARCH table it is not a requirement that every value is represented in the table, those values of choice can be included into the SUBSEARCH table.

As noted above, a search can then be performed using, for example the search methods disclosed in PCT/AU95/00560, but in the present invention, the filter uses the SUBSEARCH table instead of the SEARCH table, and may, for example, add to the SQL a clause of the form 'AND CID = n' to address the particular component.

Although the present invention is disclosed with reference to PCT/AU95/00560, the invention should not be limited to the system and methods disclosed therein. As will be understood from a reading of the specification as a whole, the present invention may be applied or implemented in a number of different systems or use a variety of methods. Nonetheless, without limitation, one preferred implementation of the present

invention is in the system as disclosed in PCT/AU95/00560, and thus the description of the present invention made in reference to that disclosure enhances the illustration and understanding of the present invention.

Preferred embodiments of the present invention will now be described with reference to the accompanying drawings, in which

Figures 1A and 1B illustrate table structure and system design of PCT/AU95/00560, but now including a SUBSEARCH table,

Figure 2 illustrates schematically a search method of PCT/AU95/00560,

Figure 3 illustrates schematically a search method according to the present invention,

Figure 4 illustrates an example X.509 certificate and an example, in schematic form, of storing the information in a SEARCH and SUBSEARCH table, and

Figure 5 illustrates an example phone number, and an example, in schematic form, of storing the information in a SEARCH and SUBSEARCH table.

Referring to Figure 1A, the table structure/arrangement as disclosed in PCT/AU95/00560 is shown, but now including an additional table 1 which can be used in searching using the CID (component ID) 2. In the implementation shown, and bearing in mind that this is only an example implementation of the present invention and considering the above disclosure, the SUBSEARCH table may contain rows of the form (EID, AID, VID, CID, Norm), where CID 2 identifies a component of the value. Of course, the table may contain other rows, but the present invention is directed to the provision of the component identifier, and thus the description will be limited to this feature.

Table 1 as shown is additional to the 'conceptual design' of Figure 1A of PCT/AU95/00560 and is only one embodiment of the present invention. In a practical application, it is thought that the embodiments described below as relating to the 'logical design' and 'physical design' of Figure 1B are more likely to be used. Any one, or a combination of the embodiments may be used in practice, bearing in mind that the present invention is directed more to

the provision and /or use of component ID, rather than the actual table design disclosed which is only exemplary.

As noted above, a search can then be performed using, for example the search methods disclosed in PCT/AU95/00560. In the present invention, however, the filter uses the SUBSEARCH table instead of the SEARCH table, and may, for example, add to the SQL a clause of the form 'AND CID = n' to address the particular component. The advantages of using the SUBSEARCH table will be explained later.

Referring to Figure 1B, the table structure of the logical design 3 and the physical design 4 of PCT/AU95/00560 is illustrated but now including a suggested (only) SUBSEARCH table 5. In fact, it is contemplated that the SUBSEARCH table can be configured to include any data/component as selected or desired. The CID field 6 serves as a useful tool in searching components of data types. In the embodiment shown, both the SEARCH and SUBSEARCH table is provided, but this is not essential. It is, however, preferred to have both tables as this provides a choice in what is to be searched, the data type, or one or more of its components.

To gain an understanding of the manner in which PCT/AU95/00560 discloses searching, and referring to Figure 2, a number of different searches are disclosed, namely base object and whole tree 7, one level 8 and subtree 9. PCT/AU95/00560 teaches that for both the base object and whole tree searches, a SEARCH table 10 is used. For a one level search 8, the DIT table 11 and SEARCH table 10 are used, and for a subtree search 9, the TREE table 12 and SEARCH table10 are used. If the data being searched is only a component of the data, the SQL may be, for example

SELECT EID{other columns} FROM SEARCH {other tables} WHERE {filter}

in certain cases the filter may be such that an index can't be used and this is considered to be relatively slow and inefficient.

In order to understand the present invention, however, we refer to Figure 3. As with PCT/AU95/00560, the base object and whole tree searches 7 may use the SUBSEARCH table 10. For the one level search 8, the DIT

table 11 and a SUBSEARCH table 13 may be used. As noted above the SUBSEARCH table includes components or may include other useful selected information of the data stored in the SEARCH table of PCT/AU95/00560. For the subtree search 9, the TREE table 12 and SUBSEARCH table 13 may be used.

In comparison to the above example, in the present invention, if the data being searched is only a component of the data, the SQL may be, for example

SELECT EID {other columns} FROM SUBSEARCH {other tables} WHERE

{filter} AND CID=x

In this case the filter which uses the component value may make use of an index and this is a much more efficient and speedy searching method.

STRUCTURED ATTRIBUTES

The present invention has a number of applications in addition to the application of the system and methods as disclosed in PCT/AU95/00560, such as in the security area where increasingly directories are being used by Certification Authorities to store X.509 certificates. These X.509 certificates are only one example of information which can be referred to as 'complex', and the invention should not be limited to only these certificates. It will be understood that the present invention in its generality has application to any form of information which has components. Advantageously, in this particular application, the present invention can be used to address the problem of how to store X.509 certificates (complex directory attributes), by finding and managing them via their serial number, expiry date, issuer, etc (i.e. one or more of their components) in a way that is considered relatively fast and efficient.

Figure 4 illustrates the application to X.509 certificates. A certificate is illustrated schematically by numeral 14. The certificate, for the purposes of illustration only shows information such as the issuer at field 15, validity information 16, serial number 17, version 18, and subject (rick harvey) 19. In the present invention, the SEARCH table would be arranged in one row as

shown at 20. In this row, there are spaces (preferably two) separating the normalised value of each component or each field of the certificate. The SEARCH table contains a normalised value representing the entire certificate. In accordance with the invention, the SUBSEARCH table is arranged with a number of rows corresponding to each component (in this example, although this is not essential to have one row per entry) by rows 22, 23, 24, 25 and 26.

For example, assume that a simple certificate simply consists of information similar to what a credit card holds such as a serial number, an expiry date, and the cardholders name. This simple certificate has three components, a number, a date and a string, respectively.

In this simplified example, the normalised value of the certificate that would be stored in the SEARCH table (assuming it is of the form: EID, AID, VID, NORM), might be.

(xx, yy, zz, "123456 20000806123000 RICK HARVEY)"

and the SUBSEARCH table would store a number of rows (assuming of the form: EID, AID, VID, CID, NORM) e.g.

(xx, yy, zz, 0, "123456")

(xx, yy, zz, 1, "20000806123000")

(xx, yy, zz, 2, "RICK HARVEY")

where xx, yy and zz are some integer corresponding to the particular table design such as EID, AID and VID, for example.

A search for a certificate that was issued to "RICK HARVEY" that only utilised the SEARCH table would contain SQL similar to

SELECT ... FROM ... SEARCH ... WHERE AID = 27 and NORM LIKE '%RICK HARVEY%'

This type of search is considered to be poor because it will have poor performance because the query is not used and there is no guarantee that the sub string being searched for is exactly the required component.

A search for a certificate that was issued to "RICK HARVEY" utilising the SUBSEARCH table would, when applying the filter, contain the SQL similar to

SELECT ... FROM ... SUBSEARCH ... WHERE ....AID = 27 AND CID = 2 AND NORM = 'RICK HARVEY'

In the above example, we search using CID=2, because we know from the manner in which the example SUBSEARCH table is designed, that CID=2 is a 'string' representing card holders name. In this manner, CID=2, AND searching for NORM = 'RICK HARVEY' will return the entry for Rick Harvey's certificate. This query is considered to be better because it can make use of an appropriate index making it potentially fast and there is certainty that the component being searched for will find the correct entry(s).

The actual designation of characters/letter or numerals in the SUBSEARCH table design is arbitrary, and may be designed in whatever manner to suit the particular application. In Figure 4, a CID = 4, will return Rick Harvey.

Table 21, illustrates an alternative and further inventive feature, where the SEARCH table may be arranged including a check sum or finger print. This may also be included in the SEARCH table (not shown). Because the SEARCH table is only used for exact matching, the value in the SEARCH table may be a fingerprint or checksum value. This makes the storage in the SEARCH table more efficient, as there is less data required to be stored.

STRING ATTRIBUTES

The present invention also has general applicability to apparently non-complex data types such as string data types such as a multi-word sentence, or multi-line paragraph of text, or a multi-line postal address. In this case, the SUBSEARCH table is accessed, but the CID may not be explicitly provided.

For example an attribute value that is a simple sentence may be stored in a single row in the SEARCH table as

(1122, 33, 0, "MANY WORD SENTENCE")

where columns are defined as (EID, AID, VID, NORM).

In PCT/AU95/00560, if we were looking for a portion of the string data type, such as just 'WORD', this row would have been searched by looking for "%WORD%, This is considered a relatively slow search.

For the purposes of example, the words could be stored in the SUBSEARCH table as follows.

(xx, yy, zz, 0, "MANY")

(xx, yy, zz, 1, "WORD")

(xx, yy, zz, 2, "SENTENCE")

where the columns are defined as (EID, AID, VID, CID, NORM).

In this later example, the filter that is then applied would then use the SUBSEARCH table instead of the SEARCH table and have in its SQL something like

SELECT ... FROM SUBSEARCH ... WHERE AID = 33 AND NORM = "WORD"

The result would be a much faster search as we are looking for an exact match of "WORD" rather than, as above, looking for a partword "%WORD%".

ALTERNATIVE INDEX

The present invention also has general applicability to the problem of being able to add another type of index to a given attribute for the purpose of increasing performance for certain types of queries. This index, in effect, gives a different path in order to find an attribute, such as for example, reverse indexing. In this case, there may only be one component in the SUBSEARCH table. The component in effect represents an alternate form of that attribute value.

In particular, the SUBSEARCH table can cope with the problem of "ends in" searches by storing a reversed form of the value and thereby giving effect to having a reversed index on the attribute.

For example, referring to Figure 5, the telephone number 27 is entered into the SEARCH table 28 in its, as it were, normal form, and in this aspect of invention, the phone number is also entered into the SUBSEARCH table in a reverse form. Of course this aspect of invention is not limited to the reverse form, it may be any other suitable form of data entry suitable for a given situation or search, such treating the area code of a telephone number as a separate component.

When searching for a telephone extension, which is the end portion of a telephone number, the search may be expressed as a string search for "*1234" (a star being a wild card). If only the SEARCH table was to be used, then the performance of prior art systems or methods is generally poor because indexes are only possible in the metadata design for "begins with" or "exact" searches. However, if the attribute is also stored in the SUBSEARCH table in reverse, then the SUBSEARCH table could be used to do an equivalent search, e.g. "4321*" which is considered to be very fast.

In this particular example, the SEARCH table might store a telephone number for a given person as

(1122, 44, 0, "98791234")

and in the SUBSEARCH table would be stored

(1122, 44, 0, 0 "43219789")

and the SQL that attempts to find a telephone number that ends in 1234 would be of the form:

SELECT ... FROM SUBSEARCH ... WHERE AID = 44 and CID=0 and NORM LIKE '4321%'

Because we are looking for an exact match for "4321%", the search is relatively fast.

14. A method, apparatus or system as herein disclosed.

DATED this 6th day of April 2000

COMPUTER ASSOCIATES THINK INC.

RCS/SH

Fig 1a.

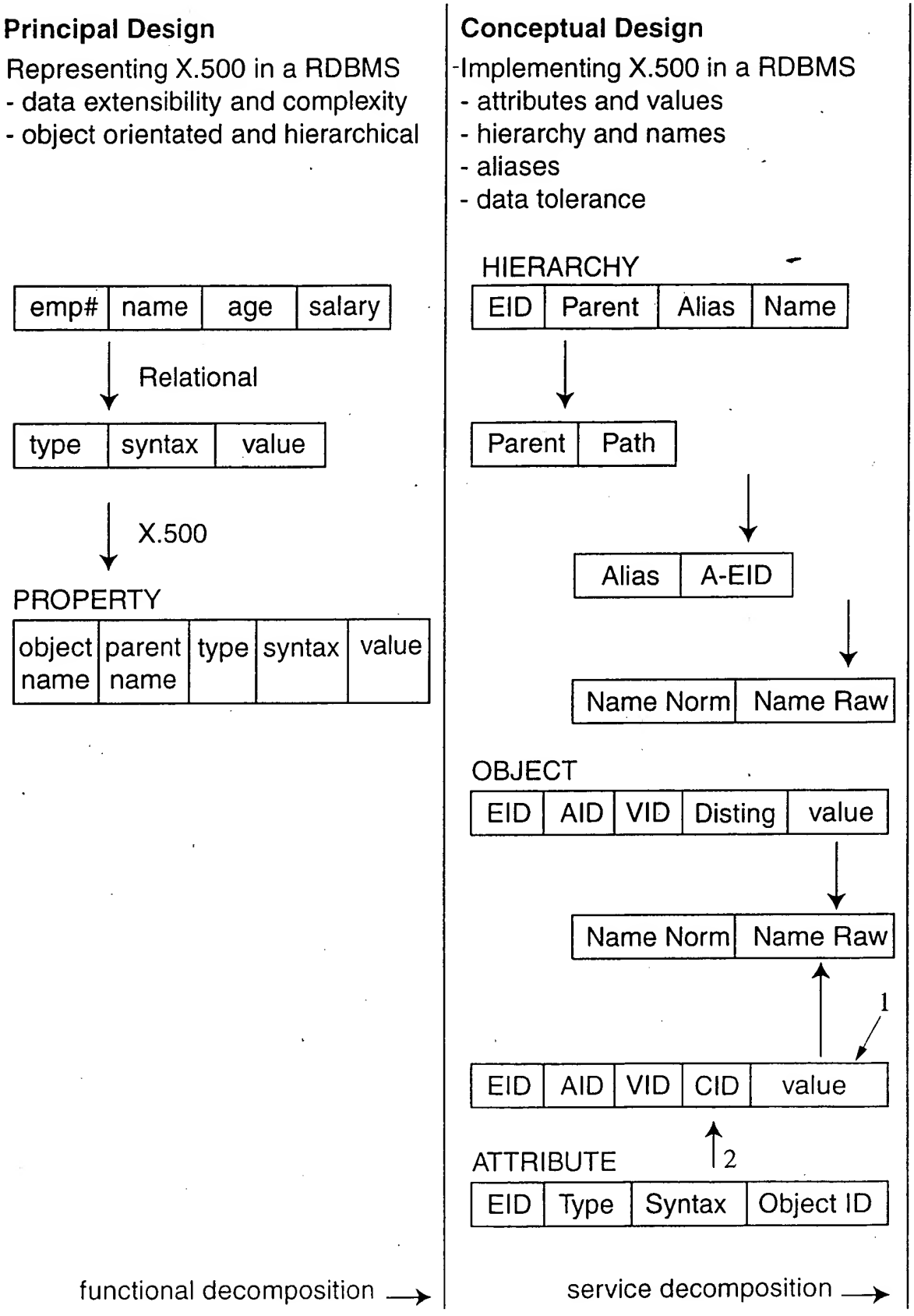

Principal Design
Representing X.500 in a RDBMS
- data extensibility and complexity
- object orientated and hierarchical
emp#
name
age
salary
Relational
type
syntax
value
X.500
PROPERTY
object name
parent name
type
syntax
value
Conceptual Design
-Implementing X.500 in a RDBMS
- attributes and values
- hierarchy and names
- aliases
- data tolerance
HIERARCHY
EID
Parent
Alias
Name
Parent
Path
Alias
A-EID
Name Norm
Name Raw
OBJECT
EID
AID
VID
Disting
value
Name Norm
Name Raw
1
EID
AID
VID
CID
value
2
ATTRIBUTE
EID
Type
Syntax
Object ID
functional decomposition
service decomposition

Fig 1b.

3

Logical Design

Performance Enhansments for RDBMS
- indexing option
- I/O considerations
- management

DIT

EID	PARENT	ALIAS	RDN

TREE

EID	PATH

ALIAS

EID	A-EID

NAME

EID	RAW

SEARCH

EID	AID	VID	DISTING	NORM

ENTRY

EID	AID	VID	RAW

SUBSEARCH

EID	AID	VID	CID	DISTING	NORM

5 6

ATTR

EID	SYX	DESC	OBJECTED

physical decomposition →

4

Physical Design

Realising X.500 in a RDBMS
- efficiency
- portability
- functional extensibility

DIT

EID	PARENT	RDNKEY	RDN	FLAGS

TREE

EID	LEV1	LEV2	LEV3	LEV4	PATH	FLAGS

ALIAS

EID	A_EID	FLAGS

NAME

EID	RAW	FLAGS

INFO

MAXIED	FLAGS

SEARCH

EID	AID	VID	NORM KEY	NORM	FLAGS

5

SUBSEARCH

EID	AID	VID	CID	NORM KEY	NORM	FLAGS

6

ENTRY

EID	AID	VID	RAW	FLAGS

SENTRY

EID	AID	VID	VALUE	FLAGS

BLOB

EID	AID	VID	VFRAG	RAW	FLAGS

ATTR

AID	SYX	DESC	OBJECT ID	FLAGS

OCLASS

OCID	DESC	OBJECTID	MUST LIST	MAY LIST	SUPER LIST	FLAGS

Fig 2.

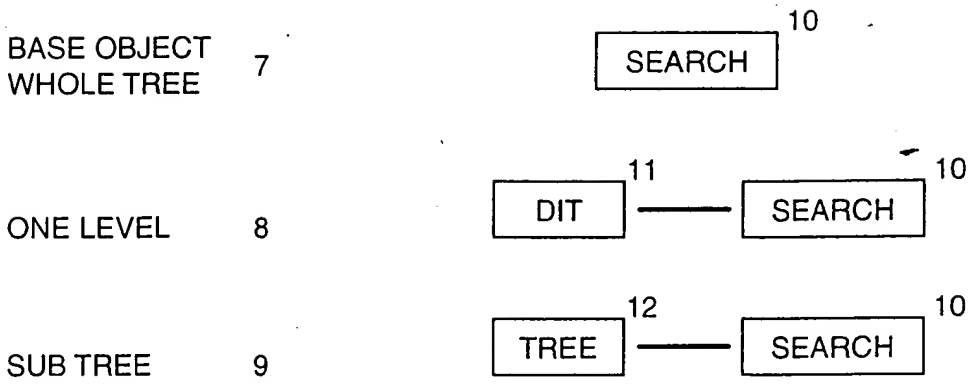

BASE OBJECT
WHOLE TREE
7
SEARCH
10
ONE LEVEL
8
DIT
11
SEARCH
10
SUB TREE
9
TREE
12
SEARCH
10

Fig 3.

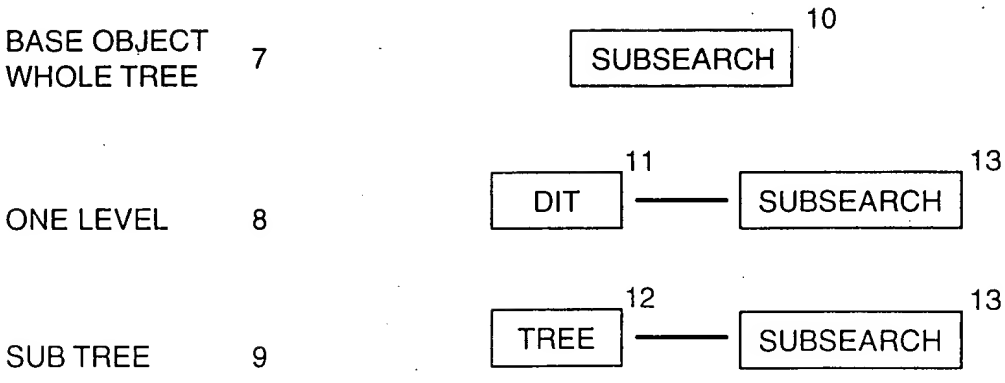

BASE OBJECT
WHOLE TREE
7
SUBSEARCH
10
ONE LEVEL
8
DIT
11
SUBSEARCH
13
SUB TREE
9
TREE
12
SUBSEARCH
13

Fig 4.

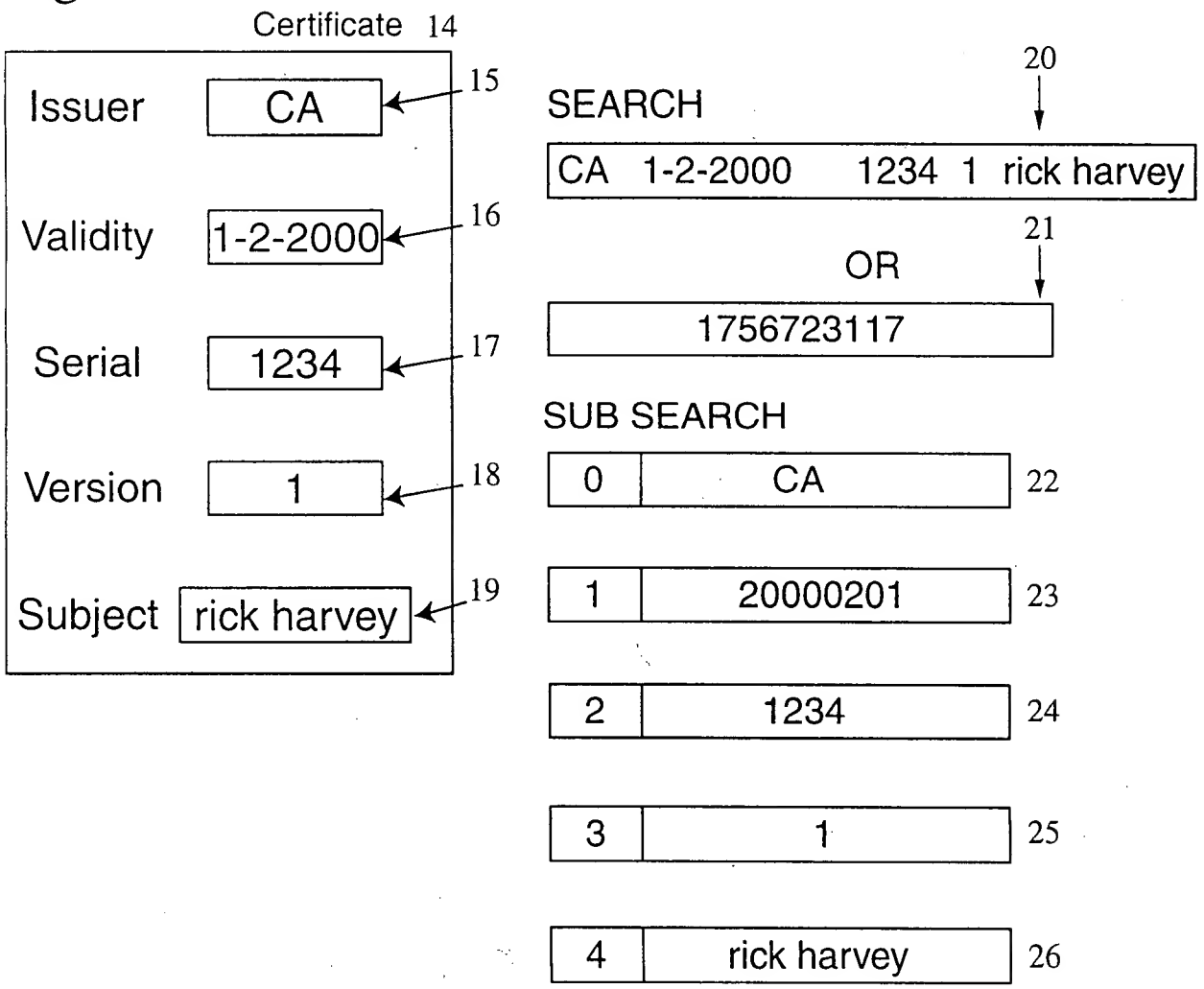
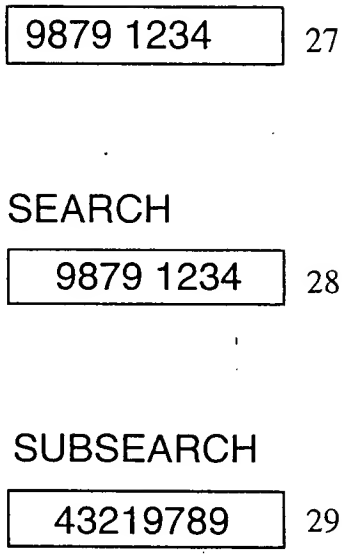

Certificate 14
Issuer
CA
15
Validity
1-2-2000
16
Serial
1234
17
Version
1
18
Subject
rick harvey
19
20
SEARCH
CA 1-2-2000 1234 1 rick harvey
21
OR
1756723117
SUB SEARCH
0
CA
22
1
20000201
23
2
1234
24
3
1
25
4
rick harvey
26

Fig 5.


phone no
9879 1234
27
SEARCH
9879 1234
28
SUBSEARCH
43219789
29